第 3 章　排版数学公式

Du Ang
du2ang233@gmail.com

2017 年 7 月 17 日

目录

本章将简单地介绍如何用 LaTeX 进行它所擅长的数学排版。如果这一章介绍的数学排版内容无法解决你的问题，不要灰心，使用 AMS-LaTeX 宏集，一般都能找到答案。

1 AMS-LaTeX 宏集

AMS-LaTeX 宏集是一些用于数学排版的包和类的集合，可以排版出高质量的数学内容。amsmath 宏包是这个宏集的核心部分，一般用这个就足够了。AMS-LaTeX 是由美国数学学会（American Mathematical Society）提供的对 LaTeX 原生的数学公式排版的扩展，目前 TeX 最近的发行版都默认包含它了。在本章，需要在导言区使用 `\usepackage{amsmath}` 命令引入 amsmath 宏包。

2 单个方程

行内公式应该放在一对 $ 之间。

示例代码：

```
Add $a$ squared and $b$ squared to get $c$ squared. Or, using
a more mathematical approach:
$a^2 + b^2 = c^2$

\TeX{} is pronounced as $\tau\epsilon\chi$ \\ [5pt]
100~m$^{3}$ of water \\ [5pt]
This comes from my $\heartsuit$
```

示例输出：

Add a squared and b squared to get c squared. Or, using a more mathematical approach: $a^2 + b^2 = c^2$

TeX is pronounced as $\tau\epsilon\chi$

100 m^3 of water

This comes from my ♡

行间公式放在 `\begin{equation}` 和 `\end{equation}` 之间。可以通过 `\label` 和 `\eqref` 来给公式添加标签和建立引用，用 `\tag` 来给公式指定具体的名字。

示例代码：

```
Add $a$ squared and $b$ squared to get $c$ squared. Or, using
a more mathematical approach:
    \begin{equation}
        a^2 + b^2 = c^2
    \end{equation}
Einstein says
    \begin{equation}
        E = mc^2 \label{clever}
```

```
    \end{equation}
He didn't say
    \begin{equation}
        1 + 1 = 3 \tag{dumb}
    \end{equation}
This a reference to \eqref{clever}.
```

示例输出：

Add a squared and b squared to get c squared. Or, using a more mathematical approach:

$$a^2 + b^2 = c^2 \tag{1}$$

Einstein says

$$E = mc^2 \tag{2}$$

He didn't say

$$1 + 1 = 3 \tag{dumb}$$

This a reference to (2).

如果不想给公式编号，用 `equation` 的加星版本 `equation*`；或者把公式放在 `\[` 和 `\]` 之间。
示例代码：

```
Add $a$ squared and $b$ squared to get $c$ squared. Or, using a more mathematical approach
    \begin{equation*}
        a^2 + b^2 = c^2
    \end{equation*}
or you can type less for the same effect:
    \[ a^2 + b^2 = c^2 \]
```

示例输出：

Add a squared and b squared to get c squared. Or, using a more mathematical approach

$$a^2 + b^2 = c^2$$

or you can type less for the same effect:

$$a^2 + b^2 = c^2$$

虽然 `\[` 和 `\]` 很简洁，但是使用时不能像 `equation` 和 `equation*` 那样在有编号和无编号之间切换。

注意排版格式中行内公式（text style）和行间公式（display style）的区别。
示例代码：

```
This is text style:
$\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$.
```

```
And this is display style:
    \begin{equation}
        \lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}
    \end{equation}
```

示例输出：

This is text style: $\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6}$.

And this is display style:

$$\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6} \tag{3}$$

在排版行间公式时，可以把一些比较高的公式放在 `\smash` 命令里，让 LaTeX 忽略这些公式的高度，使行间距保持不变。

示例代码：

```
A $d_{e_{e_{p_{e_r}}}}$ mathematical expression followed \\
by a $h^{i^{g^{h^{e^r}}}}$ expression. As opposed to a \\
smashed \smash{$d_{e_{e_{p_{e_r}}}}$} expression followed by a
\smash{$h^{i^{g^{h^{e^r}}}}$} \\ expression.
```

示例输出：

A $d_{e_{e_{p_{e_r}}}}$ mathematical expression followed
by a $h^{i^{g^{h^{e^r}}}}$ expression. As opposed to a
smashed $d_{e_{e_{p_{e_r}}}}$ expression followed by a $h^{i^{g^{h^{e^r}}}}$
expression.

2.1 数学模式（Math Mode）

当使用 `$` 开启行内公式输入，或是使用 `\[` 命令、`equation` 环境时，就进入了数学模式。数学模式相比于文本模式（text mode）有以下特点:

1. 大多数的空格、换行都不起作用，数学符号的间隙默认完全由符号的性质（关系符号、运算符等）决定。需要人为引入空隙时，使用 `\,`、`\quad` 和 `\qquad` 等命令
2. 不允许有空行、分段
3. 所有的字母被当作公式中的变量处理，字母间距与文本模式不一致，也无法生成单词间的空格。如果想在数学公式中输入正体的文本，可以使用 `\text{...}` 命令

示例代码：

```
$\forall x \in \mathbf{R}: \qquad x^{2} \geq 0$


$x^{2} \geq 0\qquad \text{for all }x\in\mathbf{R}$
```

示例输出：

$\forall x \in \mathbf{R}: \qquad x^2 \geq 0$

$x^2 \geq 0 \qquad \text{for all } x \in \mathbf{R}$

数学家们对应该使用什么符号很挑剔：上面的公式最好使用“blackboard bold”字体，可以使用 amssymb 宏包里的 \mathbb 命令来完成。

示例代码：

```
$x^{2} \geq 0\qquad \text{for all } x \in \mathbb{R}$
```

示例输出：

$x^2 \geq 0 \qquad \text{for all } x \in \mathbb{R}$

3 构建数学公式块

在这一小节中大部分的命令都不需要引入 amsmath 宏包。

3.1 希腊字母

小写希腊字母通过 \alpha、\beta、\gamma 等输入，大写希腊字母通过 \Gamma、\Delta 等输入。

示例代码：

```
$\lambda, \xi, \pi, \theta, \mu, \Phi, \Omega, \Delta$
```

示例输出：

$\lambda, \xi, \pi, \theta, \mu, \Phi, \Omega, \Delta$

3.2 指数（Exponents），上标（Superscripts），下标（Subscripts）

指数、上标和下标可以分别通过 ^ 和 _ 指定。大多数的数学模式命令仅对它之后的那个字母起作用，所以如果想对多个字母起作用，应该用 {...} 括起来。

示例代码：

```
$p^3_{ij} \qquad m_\text{Knuth}\qquad \sum_{k=1}^3 k \\[5pt]
a^x+y \neq a^{x+y}\qquad e^{x^2} \neq {e^x}^2$
```

示例输出：

$p^3_{ij} \qquad m_\text{Knuth}\qquad \sum_{k=1}^3 k$

$a^x + y \neq a^{x+y} \qquad e^{x^2} \neq e^{x2}$

3.3 根式

通过 \sqrt 输入平方根（square root）；通过 \sqrt[n] 输入 n 次方根。根号的大小是由 LaTeX 自动决定的，如果只需要一个符号标记，可以用 \surd 命令。

示例代码：

```
$\sqrt{x} \Leftrightarrow x^{1/2} \quad \sqrt[3]{2} \quad \sqrt{x^{2} + \sqrt{y}}
\quad \surd[x^2 + y^2]$
```

示例输出：

$$\sqrt{x} \Leftrightarrow x^{1/2} \quad \sqrt[3]{2} \quad \sqrt{x^{2} + \sqrt{y}} \quad \surd[x^2 + y^2]$$

3.4 点

示例代码：

```
$\Psi = v_1 \cdot v_2 \cdot \ldots \qquad n! = 1 \cdot 2 \cdots (n-1) \cdot n
\qquad \vdots \qquad \ddots$
```

示例输出：

$$\Psi = v_1 \cdot v_2 \cdot \ldots \qquad n! = 1 \cdot 2 \cdots (n-1) \cdot n \qquad \vdots \qquad \ddots$$

3.5 横线

利用 `\overline` 和 `\underline` 命令来产生上划线和下划线。
示例代码：

```
$0.\overline{3} = \underline{\underline{1/3}}$
```

示例输出：

$$0.\overline{3} = \underline{\underline{1/3}}$$

3.6 水平花括号

利用 `\overbrace` 和 `\underbrace` 命令来产生水平的上、下花括号。
示例代码：

```
$\underbrace{\overbrace{a+b+c}^6 \cdot \overbrace{d+e+f}^7}_\text{meaning of life} = 42$
```

示例输出：

$$\underbrace{\overbrace{a+b+c}^{6} \cdot \overbrace{d+e+f}^{7}}_{\text{meaning of life}} = 42$$

3.7 数学重音符号

示例代码：

```
$f(x) = x^2 \qquad f'(x) = 2x \qquad f''(x) = 2\\[5pt]
\hat{XY} \quad \widehat{XY} \quad \bar{x_0} \quad \bar{x}_0\\[5pt]
\tilde{a} \quad \widetilde{a}$
```

注意：上面示例代码中，`\\` 命令后跟了可选参数 `[5pt]` 来增加额外的行距。
示例输出：

$f(x) = x^2 \qquad f'(x) = 2x \qquad f''(x) = 2$

$\hat{XY} \quad \widehat{XY} \quad \bar{x_0} \quad \bar{x}_0$

$\tilde{a} \quad \widetilde{a}$

3.8 向量

在变量上加箭头来表示向量，可以通过 `\vec` 命令完成。

示例代码：

```
$\vec{a} \qquad \vec{AB} \qquad \overrightarrow{AB}$
```

示例输出：

$\vec{a} \qquad \vec{AB} \qquad \overrightarrow{AB}$

3.9 函数名

在排版数学公式时，通常函数名不像变量那样用斜体，所以要用命令来进行区别。下面是 LaTeX 中常用的函数名命令：

```
\arccos  \cos   \csc  \exp  \ker     \limsup
\arcsin  \cosh  \deg  \gcd  \lg      \ln
\arctan  \cot   \det  \hom  \lim     \log
\arg     \coth  \dim  \inf  \liminf  \max
\sinh    \sup   \tan  \tanh \min     \Pr
\sec     \sin
```

对于上面没有列出的一些函数，可以在导言区用 `\DeclareMathOperator` 或其加星版本的命令来定义。加星和不加星版本主要是在显示为行内公式还是行间公式上有区别。[1]

示例代码：

```
\begin{equation*}
    \lim_{x \rightarrow 0} \frac{\sin x}{x} = 1
\end{equation*}

% \DeclareMathOperator{\argh}{argh}
% \DeclareMathOperator*{\nut}{Nut}
\begin{equation*}
    3\argh = 2\nut_{x=1}
\end{equation*}
```

示例输出：

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1$$

$$3\operatorname{argh} = 2\operatorname*{Nut}_{x=1}$$

[1] MathJax: \DeclareMathOperator and \operatorname vs starred versions: http://mathb.in/13571

3.10 取模函数

取模函数有两种命令：\bmod 和 \pmod。

示例代码：

```
$a\bmod b \\
 x\equiv a \pmod{b}$
```

示例输出：

$a \bmod b$

$x \equiv a \pmod{b}$

3.11 分式和偏导

可以通过 \frac{...}{...} 命令来排版分式。在行内公式中，分式会收缩以和所在行保持一致。可以通过 \tfrac 命令在行间公式中将分式强制显示为行内模式，同样地，也可以通过 \dfrac 命令在行内公式中将分式强制显示为行间模式。一般，在行间公式中更经常使用斜杠的形式来表示比较短的分式，如 1/2。

示例代码：

```
In display style:
\begin{equation*}
    3/8 \qquad \frac{3}{8} \qquad \tfrac{3}{8}
\end{equation*}

In text style:
$1\frac{1}{2}$~hours \qquad $1\dfrac{1}{2}$~hours
```

示例输出：

In display style:

$$3/8 \qquad \frac{3}{8} \qquad \tfrac{3}{8}$$

In text style: $1\frac{1}{2}$ hours $1\dfrac{1}{2}$ hours

使用 \partial 命令来输入偏导符号。

示例代码：

```
\begin{equation*}
    \sqrt{\frac{x^2}{k+1}} \qquad x^\frac{2}{k+1} \qquad
    \frac{\partial^2f}{\partial x^2}
\end{equation*}
```

示例输出：

$$\sqrt{\frac{x^2}{k+1}} \qquad x^{\frac{2}{k+1}} \qquad \frac{\partial^2 f}{\partial x^2}$$

3.12 二项式系数和符号堆叠

可以通过 amsmath 宏包里的 \binom 命令来排版二项式系数（binomial coefficients）。
示例代码：

```
Pascal's rule is
\begin{equation*}
    \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}
\end{equation*}
```

示例输出：

Pascal's rule is

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

对于一些二元关系，有时候可能需要对符号进行堆叠，这时可以使用 \stackrel{#1}{#2} 命令。其中 #1 会变成上标大小，#2 会保持原来的大小和位置。
示例代码：

```
\begin{equation*}
    f_n(x) \stackrel{*}{\approx} 1
\end{equation*}
```

示例输出：

$$f_n(x) \stackrel{*}{\approx} 1$$

3.13 积分运算符、求和运算符、乘积运算符

分别用 \int、\sum、\prod 命令来输入积分运算符、求和运算符和乘积运算符。
示例代码：

```
\begin{equation*}
    \sum_{i=1}^n \qquad \int_0^{\frac{\pi}{2}} \qquad \prod_\epsilon
\end{equation*}
```

示例输出：

$$\sum_{i=1}^{n} \qquad \int_{0}^{\frac{\pi}{2}} \qquad \prod_{\epsilon}$$

在更复杂的表达式中，可以通过 amsmath 宏包中的 \substack 命令来更好地控制索引的位置。
示例代码：

```
\begin{equation*}
    \sum^n_{\substack{0<i<n \\ j\subseteq i}} P(i,j) = Q(i,j)
\end{equation*}
```

示例输出：

$$\sum_{\substack{0<i<n\\ j\subseteq i}}^{n} P(i,j) = Q(i,j)$$

LaTeX 可以输入各种括号（bracketing）和分隔符（delimiters）。圆括号和方括号可以通过各自的键位输入，花括号需要使用 \{ 命令，其他的分隔符都要用特殊的命令（如 \updownarrow）。

示例代码：

```
\begin{equation*}
    {a,b,c} \neq \{a,b,c\}
\end{equation*}
```

示例输出：

$$a,b,c \neq \{a,b,c\}$$

把 \left 命令放在左分隔符前面，把 \right 放在右分隔符前面，这样 LaTeX 就会自动地调整分隔符的大小。如果不想要右半个分隔符，使用不可见的 \right. 命令。

示例代码：

```
\begin{equation*}
    1 + \left(\frac{1}{1-x^{2}} \right)^3 \qquad
    \left. \ddagger \frac{~}{~}\right)
\end{equation*}
```

示例输出：

$$1 + \left(\frac{1}{1-x^2}\right)^3 \qquad \ddagger\text{-}\Big)$$

在有些情况下，需要通过 \big、\Big、\bigg、\Bigg 来手动指定分隔符的大小。

示例代码：

```
$\Big((x+1)(x-1)\Big)^{2}$ \\
$\big( \Big( \bigg( \Bigg( \quad
\big\} \Big\} \bigg\} \Bigg\} \quad
\big\| \Big\| \bigg\| \Bigg\| \quad
\big\Downarrow \Big\Downarrow
\bigg\Downarrow \Bigg\Downarrow$
```

示例输出：

$$\Big((x+1)(x-1)\Big)^2$$

$$\big(\Big(\bigg(\Bigg(\quad \big\} \Big\} \bigg\} \Bigg\} \quad \big\| \Big\| \bigg\| \Bigg\| \quad \big\Downarrow \Big\Downarrow \bigg\Downarrow \Bigg\Downarrow$$

4 长公式折行

有时候一个等式或方程太长了，有必要使其折行。但是折行会降低等式的可读性，为了提高可读性，在折行时可遵循以下规则：

1. 应该在等号或者某个操作符前折行
2. 优先在等号前而不是操作符前折行
3. 优先在加号、减号而不是乘号前折行
4. 避免其他的折行方式

实现这种折行的最简单方式是使用 `amsmath` 宏包中的 `multline` 环境。
示例代码：

```
\begin{multline}
    a + b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q
    \\
    = r + s + t + u + v + w + x + y + z
\end{multline}
```

示例输出：

$$a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q \\ = r+s+t+u+v+w+x+y+z \quad (4)$$

与之相比，`equation` 环境可能会引入任意的一个或多个折行。而使用 `multline` 环境时，仅在需要折行的地方使用 `\\` 命令进行折行。和 `equation*` 类似，`multline*` 不对公式进行编号。
示例代码：

```
\begin{equation}
    a = b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q
    + r + s + t + u + v + w + x + y + z + aa + bb + cc + dd + ee + ff
    \label{eq:equation_too_long}
\end{equation}

\begin{multline}
    a = b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q \\
    + r + s + t + u + v + w + x + y + z + aa + bb + cc + dd + ee + ff
    \label{eq:equation_too_long_multl}
\end{multline}
```

示例输出：

$$a = b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+x+y+z+aa+bb+cc+dd+ee+ff \tag{5}$$

$$\begin{aligned} a = b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q \\ + r + s + t + u + v + w + x + y + z + aa + bb + cc + dd + ee + ff \end{aligned} \tag{6}$$

比较公式 5 和公式 6，由于等号右边很长，公式 6 的输出更好一些，但是违反了等号优先于加号、减号的规则。更好的解决方案是使用 `IEEEeqnarray` 环境，将在第 5 节讨论。

5 多行公式

在遇到一些公式中有连等的情况时，我们希望在竖直方向上等号能够对齐。为了应对这个问题，先看一些常用的方法以及它们的不足之处。

5.1 传统命令存在的问题

可以使用 `align` 环境来实现等号的对齐。

示例代码：

```
\begin{align}
    a & = b + c \\
    & = d + e
\end{align}
```

示例输出：

$$\begin{aligned} a &= b + c \\ &= d + e \end{aligned} \tag{7}\tag{8}$$

但是在某一行公式过长时，使用 `align` 环境的输出会出现问题。如下面的示例，`+ v` 应该和 `d` 对齐，而不是和上面的等号对齐。可以通过在它前面加一些空格（`\hspace{...}`）来实现我们想要的效果，但是这不够准确。

示例代码：

```
\begin{align}
    a & = b + c \\
    & = d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t + u \nonumber \\
    & + v + w + x + y + z + aa + bb + cc \\
    & = dd + ee + ff + gg + hh + ii
\end{align}
```

示例输出：

$$
\begin{aligned}
a &= b + c && (9)\\
&= d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t + u \\
&\quad + v + w + x + y + z + aa + bb + cc && (10)\\
&= dd + ee + ff + gg + hh + ii && (11)
\end{aligned}
$$

eqnarray 环境提供了一种更好的解决方案。

示例代码：

```
\begin{eqnarray}
    a & = & b + c \\
    & = & d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t + u \nonumber \\
    && +\: v + w + x + y + z + aa + bb + cc \\
    & = & dd + ee + ff + gg + hh + ii
\end{eqnarray}
```

示例输出：

$$
\begin{aligned}
a &= b + c && (12)\\
&= d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t + u \\
&\quad + v + w + x + y + z + aa + bb + cc && (13)\\
&= dd + ee + ff + gg + hh + ii && (14)
\end{aligned}
$$

但使用 eqnarray 环境仍不是最优的解决方案。等号两边的空太大了，而且和 multline 环境、equation 环境中的大小不一样。而且有些时候即使左边有足够的空间，右边还是会和公式编号重叠。

示例代码：

```
\begin{eqnarray}
    a & = & a = a
\end{eqnarray}

\begin{eqnarray}
    a & = & b + c
    \\
    & = & d + e + f + g + h^2 + i^2 + j + k + l + m + n + o + p + q + r + s + t + u + v
     + w + x + y + z
    \label{eq:faultyeqnarray}
\end{eqnarray}
```

示例输出：

$$a = a = a \tag{15}$$

$$\begin{aligned} a &= b + c && (16)\\ &= d + e + f + g + h^2 + i^2 + j + k + l + m + n + o + p + q + r + s + t + u + v + w + x + y && (17) \end{aligned}$$

eqnarray 环境提供了 \lefteqn 命令，可以在公式的等号左边部分过长时使用。但是这样并不是最优的解决方案，而且等号右边部分过短时公式可能不会居中。

示例代码：

```
\begin{eqnarray}
    \lefteqn{a + b + c + d + e + f + g + h} \nonumber \\
    & = & i + j + k + l + m \\
    & = & n + o + p + q + r + s
\end{eqnarray}

\begin{eqnarray}
    \lefteqn{a + b + c + d + e + f + g + h} \nonumber \\
    & = & i + j
\end{eqnarray}
```

示例输出：

$$\begin{aligned} a + b + c + d + e + f + g + h & \\ &= i + j + k + l + m && (18)\\ &= n + o + p + q + r + s && (19) \end{aligned}$$

$$\begin{aligned} a + b + c + d + e + f + g + h & \\ &= i + j && (20) \end{aligned}$$

5.2 IEEEeqnarray 环境

为了使用 IEEEeqnarray 环境，需要在导言区使用一下命令引入 IEEEtrantools 宏包：\usepackage[retainorgcmds]{IEEEtrantools}。

IEEEeqnarray 的一个优点是可以指定公式排列的列数。通常情况下，指定参数为 {rCl}，表示总共有三列，第一列右对齐，中间列居中（和小写字母 c 相比，大写字母 C 还会使其左右两边留有空隙），第三列左对齐。

示例代码：

```
\begin{IEEEeqnarray}{rCl}
    a & = & b + c \\
    & = & d + e + f + g + h + i + j + k \nonumber \\
    && \negmedspace {} + l + m + n + o \\
    & = & p + q + r + s
\end{IEEEeqnarray}
```

示例输出：

$$\begin{aligned} a &= b + c && (21)\\ &= d + e + f + g + h + i + j + k \\ &\quad + l + m + n + o && (22)\\ &= p + q + r + s && (23) \end{aligned}$$

可以指定任意多列：`{c}` 表示只有居中的一列；`{rCll}` 会增加左对齐的第四列，一般用于添加注释。

5.3 IEEEeqnarray 环境的一般用法

如果某行公式会和公式编号重叠，可以在对应公式行后使用 `\IEEEeqnarraynumspace` 命令来解决。
示例代码：

```
\begin{IEEEeqnarray}{rCl}
    a & = & b + c \\
    & = & d + e + f + g + h^2 + i^2 + j + k + l + m + n + o + p + q + r + s + t + u + v
     + w + x + y + z
\end{IEEEeqnarray}

\begin{IEEEeqnarray}{rCl}
    a & = & b + c \\
    & = & d + e + f + g + h^2 + i^2 + j + k + l + m + n + o + p + q + r + s + t + u + v
     + w + x + y + z \IEEEeqnarraynumspace
\end{IEEEeqnarray}
```

示例输出：

$$\begin{aligned} a &= b + c && (24)\\ &= d + e + f + g + h^2 + i^2 + j + k + l + m + n + o + p + q + r + s + t + u + v + w + x + y + z && (25) \end{aligned}$$

$$\begin{aligned} a &= b + c && (26)\\ &= d + e + f + g + h^2 + i^2 + j + k + l + m + n + o + p + q + r + s + t + u + v + w + x + y + z && (27) \end{aligned}$$

如果等号左边部分过长，IEEEeqnarray 提供了 \IEEEeqnarraymulticol 命令来取代有缺陷的 \lefteqn 命令。

示例代码：

```
\begin{IEEEeqnarray}{rCl}
    \IEEEeqnarraymulticol{3}{l}{
        a + b + c + d + e + f + g + h
    } \nonumber \\ \quad
    & = & i + j \\
    & = & k + l + m
\end{IEEEeqnarray}

\begin{IEEEeqnarray}{rCl}
    \IEEEeqnarraymulticol{3}{l}{
        a + b + c + d + e + f + g + h
    } \nonumber \\ \qquad \qquad
    & = & i + j \\
    & = & k + l + m
\end{IEEEeqnarray}
```

示例输出：

$$\begin{aligned} & a+b+c+d+e+f+g+h \\ & \quad = i+j && (28) \\ & \quad = k+l+m && (29) \end{aligned}$$

$$\begin{aligned} & a+b+c+d+e+f+g+h \\ & \qquad\qquad = i+j && (30) \\ & \qquad\qquad = k+l+m && (31) \end{aligned}$$

\IEEEeqnarraymulticol 的用法和 tabular 环境中 \multicolumns 命令的用法是相同的。第一个参数 {3} 指定三列合并为一列，{l} 参数指定其左对齐。可以通过 \quad 等命令可以方便地控制等号的缩进。

如果公式折成了两行或多行，LaTeX 会把首个 + 或者 − 作为一个标识符，而不是运算符。因此需要在运算符和操作数之间插入额外的空格。

示例代码：

```
\begin{IEEEeqnarray}{rCl}
    a & = & b + c \\
    & = & d + e + f + g + h + i + j + k \nonumber \\
```

```
    && + l + m + n + o \\
    & = & p + q + r + s
\end{IEEEeqnarray}

\begin{IEEEeqnarray}{rCl}
    a & = & b + c \\
    & = & d + e + f + g + h + i + j + k \nonumber \\
    && \negmedspace {} + l + m + n + o \\ % negative medium space
    & = & p + q + r + s
\end{IEEEeqnarray}
```

示例输出：

$$
\begin{aligned}
a &= b + c && (32)\\
&= d + e + f + g + h + i + j + k \\
&\quad + l + m + n + o && (33)\\
&= p + q + r + s && (34)
\end{aligned}
$$

$$
\begin{aligned}
a &= b + c && (35)\\
&= d + e + f + g + h + i + j + k \\
&\quad + l + m + n + o && (36)\\
&= p + q + r + s && (37)
\end{aligned}
$$

IEEEeqnarray 环境也有加星版本，会省略所有的公式编号，但可以使用 \IEEEyesnumber 和 \IEEEyessubnumber 命令来显示某行公式的编号或子编号。

示例代码：

```
\begin{IEEEeqnarray*}{rCl}
    a & = & b + c \\
    & = & d + e \IEEEyesnumber \\
    & = & f + g
\end{IEEEeqnarray*}

\begin{IEEEeqnarray}{rCl}
    a & = & b + c \IEEEyessubnumber \\
    & = & d + e \nonumber \\
    & = & f + g \IEEEyessubnumber
\end{IEEEeqnarray}
```

示例输出：

$$\begin{aligned} a &= b + c \\ &= d + e \\ &= f + g \end{aligned} \tag{38}$$

$$a = b + c \tag{38a}$$
$$= d + e$$
$$= f + g \tag{38b}$$

6 数组和矩阵

可以通过 array 环境来排版数组，它的工作方式和 tabular 环境类似。
示例代码：

```
\begin{equation*}
    \mathbf{X} = \left(
        \begin{array}{ccc}
            x_1 & x_2 & \ldots \\
            x_3 & x_4 & \ldots \\
            \vdots & \vdots & \ddots
        \end{array} \right)
\end{equation*}
```

示例输出：

$$\mathbf{X} = \left(\begin{array}{ccc} x_1 & x_2 & \ldots \\ x_3 & x_4 & \ldots \\ \vdots & \vdots & \ddots \end{array} \right)$$

通过将 . 作为一个不可见的右分隔符，array 环境也可以用于排版分段函数（piecewise functions）。amsmath 宏包中的 cases 环境也可以实现类似的效果，且语法更为简洁。
示例代码：

```
\begin{equation*}
    |x| = \left\{
        \begin{array}{rl}
            -x & \text{if } x < 0, \\
            0 & \text{if } x = 0, \\
            x & \text{if } x > 0.
        \end{array} \right.
```

```
\end{equation*}

\begin{equation*}
    |x| =
    \begin{cases}
        -x & \text{if } x < 0, \\
        0 & \text{if } x = 0, \\
        x & \text{if } x > 0.
    \end{cases}
\end{equation*}
```

示例输出：

$$|x| = \begin{cases} -x & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ x & \text{if } x > 0. \end{cases}$$

$$|x| = \begin{cases} -x & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ x & \text{if } x > 0. \end{cases}$$

可以通过 `array` 环境来排版矩阵，但是 `amsmath` 提供了一系列的 `matrix` 环境，可以更好地排版矩阵。共有六种不同的分隔符：`matrix`（无）、`pmatrix`（()、`bmatrix`（[]、`Bmatrix`（{}、`vmatrix`（|）和 `Vmatrix`（||）。和 `array` 环境不同，`matrix` 环境不需要指定列数，最大列数为 10。

7 数学模式中的空格

如果 LaTeX 数学公式中的空格不能满足要求，可以通过插入一些特殊的命令进行调整。`\,` 命令对应 $\frac{3}{18}$ 个 quad；`\:` 命令对应 $\frac{4}{18}$ 个 quad；`\;` 命令对应 $\frac{5}{18}$ 个 quad；转移空格字符（escaped space character）`\␣` 会产生一个和单词间距大小相当的空格；`\quad` 会产生一个当前字体下'M'字母宽度的空格，`\qquad` 相当于两个 `\quad`；`\!` 命令会产生一个 $-\frac{3}{18}$ 个 quad。

示例代码：

```
\begin{equation*}
    \int_1^2 \ln x \mathrm{d}x
    \qquad
    \int_1^2 \ln x \, \mathrm{d}x
\end{equation*}
```

示例输出：

$$\int_1^2 \ln x \mathrm{d}x \qquad \int_1^2 \ln x \, \mathrm{d}x$$

注意微分运算中的字母‘d’一般用罗马字体。下面的例子会定义一个新命令 \ud（upright d），可以实现相同的效果。

示例代码：

```
\newcommand{\ud}{\, \mathrm{d}}

\begin{IEEEeqnarray*}{c}
    \int\int f(x)g(y) \ud x \ud y \\
    \int\!\!\!\int f(x)g(y) \ud x \ud y \\
    \iint f(x)g(y) \ud x \ud y
\end{IEEEeqnarray*}
```

示例输出：

$$\int\int f(x)g(y)\,\mathrm{d}x\,\mathrm{d}y$$
$$\iint f(x)g(y)\,\mathrm{d}x\,\mathrm{d}y$$
$$\iint f(x)g(y)\,\mathrm{d}x\,\mathrm{d}y$$

7.1 幽灵（Phantoms）

\phantom 命令可以为其中的内容保留位置，但这些内容又不会在最终的输出中出现。下面的例子可以帮助我们很好地理解它的用途。

示例代码：

```
\begin{equation*}
    {}^{14}_{6}\text{C}
    \qquad \text{versus} \qquad
    {}^{14}_{\phantom{1}6}\text{C}
\end{equation*}
```

示例输出：

$$^{14}_{6}\mathrm{C} \qquad \text{versus} \qquad ^{14}_{6}\mathrm{C}$$

mhchem 宏包可以更方便地输入同位素和化学方程式。

8 折腾数学字体

示例代码：

```
$\Re \qquad \mathcal{R} \qquad \mathfrak{R} \qquad \mathbb{R} \qquad $
```

示例输出：

$\mathfrak{R}$ $\mathcal{R}$ $\mathfrak{R}$ $\mathbb{R}$

上面的后两种字体命令需要用到 amssymb 或 amsfonts 宏包。

有时候需要给 LaTeX 指定合适的字体大小。在数学模式中，可以通过以下的命令进行设定：`\displaystyle`（123），`\textstyle`（123），`\scriptstyle`（123），`\scriptscriptsytle`（123）。

如果 $\sum$ 符号放在分式中，它会默认排版成文本模式，所以可以根据需要用 `\displaystyle` 进行指定。

示例代码：

```
\begin{equation*}
    P = \frac{\displaystyle{\sum_{i=1}^n (x_i-x)(y_i-y)}}
        {\displaystyle{\left[\sum_{i=1}^n(x_i-x)^2 \sum_{i=1}^n(y_i-y)^2 \right]^{1/2}}}
\end{equation*}
```

示例输出：

$$P = \frac{\displaystyle\sum_{i=1}^{n}(x_i-x)(y_i-y)}{\displaystyle\left[\sum_{i=1}^{n}(x_i-x)^2\sum_{i=1}^{n}(y_i-y)^2\right]^{1/2}}$$

8.1 粗体符号

在 LaTeX 中打印粗体符号有一些困难。`\mathbf` 命令可以打印粗体的的英文字母，但是打印出的是罗马字体的，而数学公式中的字母大都是斜体的，而且该命令对希腊字母无效。`\boldmath` 命令可以打印粗体的英文字母和希腊字母，但是它不能用在数学公式中，只能包裹在数学公式外面。

amsbsy 宏包（包含于 amsmath）和 bm 宏包（包含于 tools）中都有一个 boldsymbol 命令，可以比较方便地打印粗体符号。

示例代码：

```
$\mu, M \qquad \mathbf{\mu}, \mathbf{M}$
\qquad \boldmath{$\mu, M$}
$\qquad \boldsymbol{\mu}, \boldsymbol{M}$
```

μ, M $\mu, \mathbf{M}$ $\boldsymbol{\mu}, \boldsymbol{M}$ $\boldsymbol{\mu}, \boldsymbol{M}$

9 定理（theorems），引理（Lemmas）……

在撰写数学文档的时候，可能需要排版引理、定义、公理等。可以通过下面的命令来完成：

```
\newtheorem{name}[counter]{text}[section]
```

其中，*name* 参数是我们定义定理的名称，作为一个环境来使用；*text* 参数是定理真正的名字，会显示在最终的文档中。方括号中的参数是可选的，定理的序号由两个可选参数之一决定，它们不能同时使用。*section* 为章节名称，使定理序号成为章节的下一级序号。*counter* 可以为用 `\newcounter` 自定义的

计数器名称，定理序号由这个计数器管理；也可以为前面定义的“theorem”的名称。如果两个参数都不用的话，则使用一个默认的计数器。

例如，我们用 `\newtheorem{mythm}{My Theorem}[section]` 命令定义了一个 `mythem` 环境，然后就可以用它来排版定理。定理带有一个可选参数，可以用于注明定理的名称。

示例代码：

```
\newtheorem{mythm}{My Theorem}[section]

\begin{mythm}
    \label{thm:light}
    The light speed in vaccum is $299,792,458\,\mathrm{m/s}$.
\end{mythm}

\begin{mythm}[Energy]
    The relationship of energy, momentum and mass is
        \[E^2 = m_0^2 c^4 + p^2 c^2\]
    where $c$ is the light speed described in theorem \ref{thm:light}.
\end{mythm}
```

示例输出：

My Theorem 9.1. *The light speed in vaccum is* $299,792,458\,\mathrm{m/s}$.

My Theorem 9.2 (Energy)**.** *The relationship of energy, momentum and mass is*

$$E^2 = m_0^2 c^4 + p^2 c^2$$

where c is the light speed described in theorem 9.1.

amsthm 宏包提供了 `\theoremstyle{style}` 命令，可以定义不同的 theorem 类型，包括 `definition`、`plain`、`remark` 三种。

示例代码：

```
\theoremstyle{definition}   \newtheorem{law}{Law}
\theoremstyle{plain}        \newtheorem{jury}[law]{Jury}
\theoremstyle{remark}       \newtheorem*{marg}{Margaret}

\begin{law}
    \label{law:box}
    Don't hide in the witness box
\end{law}
\begin{jury}[The Twelve]
    It could be you! So beware and see law~\ref{law:box}.
\end{jury}
```

```
\begin{jury}
    You will disregard the last statement.
\end{jury}
\begin{marg}No, No, No \end{marg}
\begin{marg}Denis! \end{marg}
```

示例输出：

Law 1. Don't hide in the witness box

Jury 2 (The Twelve)**.** *It could be you! So beware and see law 1.*

Jury 3. *You will disregard the last statement.*

Margaret. No, No, No

Margaret. Denis!

在上面的示例中，“Jury”定理和“Law”定理用了相同的 *counter* 参数，所以“Jury”的编号是接着“Law”的。

示例代码：

```
\newtheorem{mur}{Murphy}[section]

\begin{mur}
    If there are two or more ways to do something, and one of those ways can
    result in a catastrophe, then someone will do it.
\end{mur}
```

示例输出：

Murphy 9.1. *If there are two or more ways to do something, and one of those ways can result in a catastrophe, then someone will do it.*

上面示例中的“Murphy”定理由于定义时指定了 *section* 参数，所以它的编号和当前章节有关。*section* 参数也可以换成 *chapter* 或 *subsection*。

9.1 证明（Proofs）和证毕符号（End-of-Proof Symbol）

amsthm 宏包也提供了 proof 环境。

示例代码：

```
\begin{proof}
    Trival, use
    \begin{equation*}
        E = mc^2.
```

```
    \end{equation*}
\end{proof}

\begin{proof}
    Trival, use
    \begin{equation*}
        E = mc^2. \qedhere
    \end{equation*}
\end{proof}
```

示例输出：

证明. Trival, use

$$E = mc^2.$$

□

证明. Trival, use

$$E = mc^2.$$ □

如果行末是一个不带编号的公式，证毕符号会另起一行，这时可以使用 `\qedhere` 命令将证毕符号放在公式末尾。`\qedhere` 命令对 `IEEEeqnarray` 无效。这是由于 `IEEEeqnarray` 的内部结构导致的。为了保证公式可以水平居中，`IEEEeqnarray` 在公式块两边各加了一列。这两列只包含一个可以拉伸的空格，是不可见的，因此 `\qedhere` 命令无法放在可拉伸空格的外面。

示例代码：

```
\begin{proof}
    This is a proof that ends with an equation array:
    \begin{IEEEeqnarray*}{rCl}
        a & = & b + c \\
        & = & d + e. \qedhere
    \end{IEEEeqnarray*}
\end{proof}
```

示例输出：

证明. This is a proof that ends with an equation array:

$$\begin{aligned} a &= b + c \\ &= d + e. \end{aligned}$$ □

对于上面的问题，有一种简单的解决方法，即对可拉伸的空格进行明确指定。

示例代码：

```
\begin{proof}
    This is a proof that ends with an equation array:
```

```
    \begin{IEEEeqnarray*}{+rCl+x*}
        a & = & b + c \\
        & = & d + e. & \qedhere
    \begin{IEEEeqnarray*}
\end{proof}
```

示例输出：

证明. This is a proof that ends with an equation array:

$$\begin{aligned} a &= b + c \\ &= d + e. \end{aligned}$$ □

`{+rCl+x*}` 中的 `+` 表示可拉伸的空格，在公式左右各一列。现在又在右边可拉伸的空格外加了一个空白列 `x`，这一列只有最后一行的 `\qedhere` 命令需要。最后又指定一个 `*`，表示没有空格，防止 `IEEEeqnarray` 在最右边添加额外的 `+` 空格。

对于有编号的公式，也有类似的问题和解决方法。

示例代码：

```
% Wrong with equation
\begin{proof}
    This is a proof that ends with a nubmered equation:
    \begin{equation}
        a = b + c.
    \end{equation}
\end{proof}

% Right with equation
\begin{proof}
    This is a proof that ends with a nubmered equation:
    \begin{equation}
        a = b + c. \qedhere
    \end{equation}
\end{proof}

% Wrong with IEEEeqnarray
\begin{proof}
    This is a proof that ends with a equation array:
    \begin{IEEEeqnarray}{rCl}
        a & = & b + c \\
        & = & d + e.
    \end{IEEEeqnarray}
\end{proof}
```

```
% Right with IEEEeqnarray
\begin{proof}
    This is a proof that ends with an equation array:
    \begin{IEEEeqnarray}{+rCl+x*}
        a & = & b + c \\
        & = & d + e. \\
        &&& \qedhere \nonumber
    \end{IEEEeqnarray}
\end{proof}
```

示例输出：

证明. This is a proof that ends with a nubmered equation:

$$a = b + c. \tag{39}$$

□

证明. This is a proof that ends with a nubmered equation:

$$a = b + c. \tag{40}$$

□

证明. This is a proof that ends with a equation array:

$$\begin{aligned} a &= b + c && (41)\\ &= d + e. && (42) \end{aligned}$$

□

证明. This is a proof that ends with an equation array:

$$\begin{aligned} a &= b + c && (43)\\ &= d + e. && (44) \end{aligned}$$

□